Compliance Trestle

Students: Yihao Mai, Abdulaziz Memesh
Mentors: Vikas Agarwal, Lou Degenaro, Manjiree Gadgil, Alejandro Jose Leiva Palomo

Georgia Tech
Open Source Program Office

Project Overview

- Trestle is an SDK that leverages OSCAL data format to achieve continuous compliance with a collection of tools that enables the creation, validation, and governance of compliance artifacts, all of which can be accessed in the CLI and APIs.
- Two main functions:
  - Convert compliance artifact in other formats to OSCAL format
  - Allow easy creation, manipulation, and validation of OSCAL documents

Goal

Understand the open-source project workflow through the interaction with the Trestle codebase.

Milestones

1. **Week 1-2:** Understand the key concepts of the OSCAL framework such as model layers, model schemas and their usability as well as set up the environment for Trestle.
2. **Week 3-4:** Get familiar with the codebase and start resolving simple issues such as identifying and fixing bugs and refactoring documentation while complying with the contribution guideline.
3. **Week 5-6:** Add new features to the codebase.
4. **Week 7-11:** Refactor and update outdated demos, add new demos, and assist in integration to new CICD framework.

Highlights and Accomplishments

```
        subjects = self._get_subjects(result.local_definitions)
        result.observations = self._get_result_observations(yaml_data, subjects)
        return result
+
+   def is_yaml_valid(self, yaml_data_list: List[Dict]) -> bool:
+       """Check if the YAML file has the required keys."""
+       for yaml_data in yaml_data_list:
+           if ('metadata' in yaml_data and 'labels' in yaml_data['metadata'] and
+                   ('wgpolicyk8s.io/engine' in yaml_data['metadata']['labels'] or
+                   'policy.kubernetes.io/engine' in yaml_data['metadata']['labels'])):
+               return True
+       return False
+
    def transform(self, yaml_data_list: List[Dict], ar_type: str, title: str, href: str,
                  ns: str) -> Union[Results, AssessmentResults]:
        """Transform yaml to OSCAL json."""


            with open(ipath, 'r', encoding='utf-8') as yaml_file:
                for yaml_section in yaml.safe_load_all(yaml_file):
                    yaml_data_list.append(yaml_section)
+               # only collect the ymal files that have the required keys

+               if ytoo.is_yaml_valid(yaml_data_list):
+                   results = ytoo.transform(yaml_data_list, args.ar_type,
    title=ofile.name, href=args.ap_href, ns=args.ns)
+                   write_file = pathlib.Path(ofile).open('wb')
+                   write_file.write(results)
+                   write_file.flush()
+                   write_file.close()
+                   logger.info(f'created: {opath / ofile.name}')
        except yaml.YAMLError as e:
            logger.error(e)
            raise Exception(f'Exception processing {ipath.name}')
```

- Wrote a helper function to only take in accepted YAML files without deleting other YAML files, which other demos relied on, from the folder

```
spec:
  params:
    - name: repo-url
      description: The URL of the repository
    - name: repo-name
      description: The name of the repository
  tasks:
    - name: perform-update
      taskSpec:
        steps:
          - name: update
            image: python:latest
            script: |
              #!/bin/bash
              echo "Checking if any files under catalogs/ directory have changed..."

              # Loop through the changed files
              matched_json_files=()
              for file in $(params.changedfiles); do
                if [[ $file == catalogs/* ]]; then
                  matched_json_files+=($file)
                fi
              done

              if [[ ${#matched_json_files[@]} -gt 0 ]]; then
                echo "Install trestle"
                python3 -m pip install -q --upgrade pip setuptools
                python3 -m pip install -q compliance-trestle
                python3 -m pip install -q python-semantic-release==7.31.4


                echo "Clone the repository"
                git clone $(params.repo-url)
                cd $(params.repo-name)
                git checkout develop
                git config --global credential.helper store
                echo "https://${TOKEN}:@github.com" > ~/.git-credentials
                git config --global user.email "yihaomai@gmail.com"
                git config --global user.name "Mathi"

                echo "Files under catalogs/ directory have changed. Files include: ${matched_json_files[@]}"
                echo "Generate the markdown files from catalog JSON file."
                trestle author catalog-generate -n . --output md_catalogs/catalog-folder
                git add .
                git commit -m "Update catalog markdown files"
                git push
                echo "Proceeding to update downstream repositories..."
              else
                echo "No files under catalogs/ directory have changed."
              fi
            env:
              - name: TOKEN
                valueFrom:
```

- One of the Tekton pipeline component's configuration files

Highlights and Accomplishments

```
@pytest.fixture(scope='session', autouse=True)
def clean_tmp():
    """Remove Trestle workspace in /tmp if there's one"""
    if os.name == 'posix':
        curr_path = os.getcwd()
        os.chdir("/tmp")
        try:
            shutil.rmtree(".trestle")
            shutil.rmtree("dist")
            shutil.rmtree("catalogs")
            shutil.rmtree("profiles")
            shutil.rmtree("component-definitions")
            shutil.rmtree("system-security-plans")
            shutil.rmtree("assessment-plans")
            shutil.rmtree("assessment-results")
            shutil.rmtree("plan-of-action-and-milestones")
        finally:
            os.chdir(curr_path)
```

Wrote a function that deletes folders that cause an error before testing

- Learning about how a CLI is created in Python
- Getting hands-on experience in working with large-scale projects

Learning Outcomes

- Gaining more familiarity with the open-source contribution workflow while navigating version control
- Understanding the process of unit testing and software testing in general
- Getting hands-on experience in improving documentation
- Learning and implementing CICD pipelines with frameworks such as GitHub Actions and Tekton

Future Work

- Improve the documentation more
- Resolve small issues and bug fixes
- Add new demos
- Implement the downstream-update feature in the Tekton pipeline.
- Integrate the Tekton pipelines to the Profile, and Component Definition template repositories.

Georgia Tech